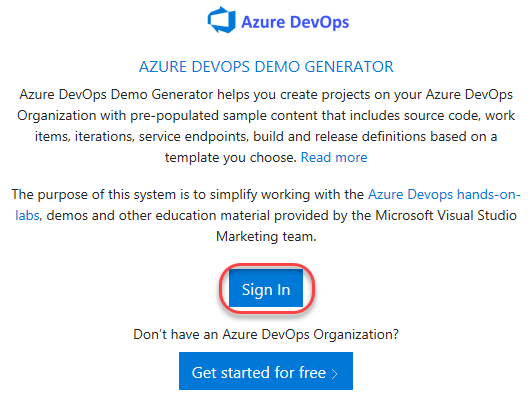
**Azure Artifacts - Demo**

**Overview**

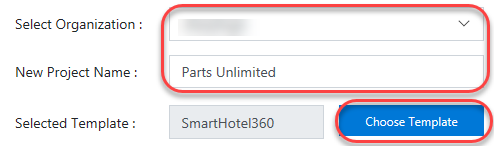
Certain Azure DevOps labs require a preconfigured **Parts Unlimited** team project. This document outlines the required steps to set up the required data.

***Task 1: Configuring the Parts Unlimited team project***

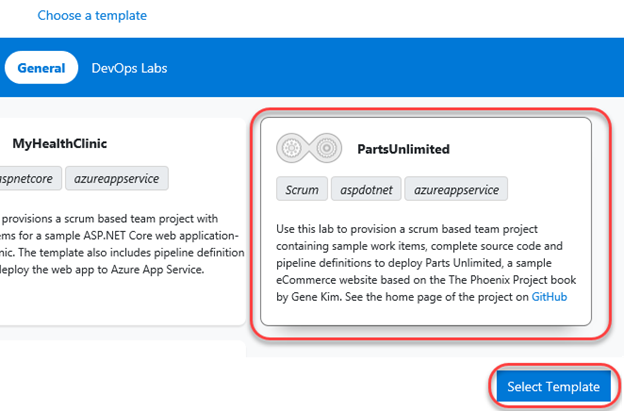
1. Navigate to [https://azuredevopsdemogenerator.azurewebsites.net](https://azuredevopsdemogenerator.azurewebsites.net/). This utility site will automate the process of creating a new Azure DevOps project within your account that is prepopulated with content (work items, repos, etc.) required for the lab. For more information on the site, please see <https://docs.microsoft.com/en-us/azure/devops/demo-gen>.
2. Sign in using the Microsoft account associated with your Azure DevOps subscription.



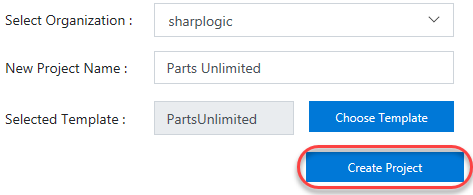
1. **Accept** the permission requests for accessing your subscription.
2. Select your Azure DevOps organization and enter the project name **“Parts Unlimited”**. Click **Choose Template**.



1. Select the **PartsUnlimited** template and click **Select Template**.

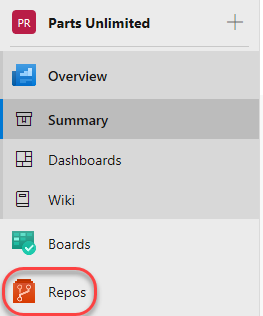


1. Click **Create Project** and wait for the process to complete.

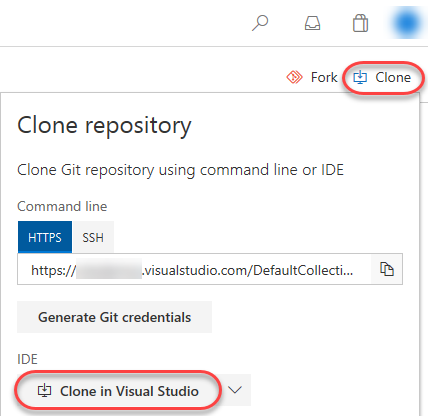


***Task 2: Configuring the Parts Unlimited solution in Visual Studio***

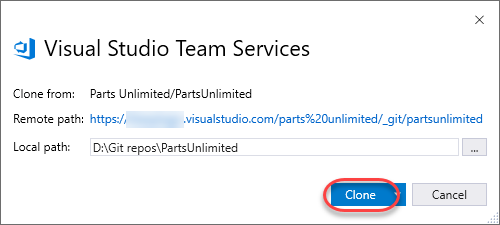
1. Some labs will require you to open the **Parts Unlimited** solution in **Visual Studio**. If your lab doesn’t require this, you can skip this task.
2. Navigate to your Azure DevOps team project for **Parts Unlimited**. It will be something like <https://dev.azure.com/YOURACCOUNT/Parts%20Unlimited>.
3. Navigate to the **Repos** hub.



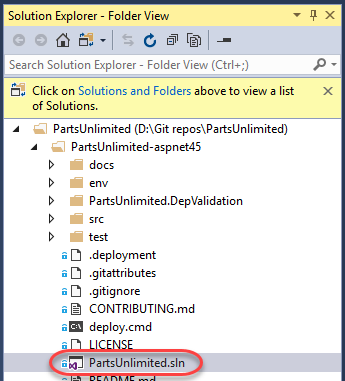
1. Click **Clone** and select **Clone in Visual Studio**.



1. Follow the workflow to clone and configure the project in Visual Studio. Click **Clone** to copy the repo locally.



1. From **Solution Explorer**, double-click **PartsUnlimited.sln** to open the solution. You can ignore warnings about unsupported project types.



1. Leave Visual Studio open for use in your lab.

## Overview

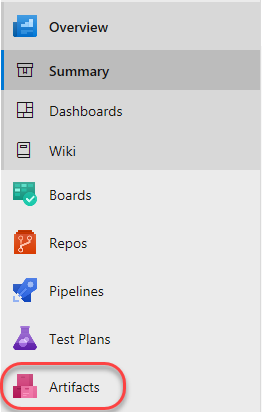
**Azure Artifacts** is an extension that makes it easy to discover, install and publish NuGet, npm and Maven packages in Azure DevOps. It’s deeply integrated with other hubs like Build so that package management can become a seamless part of your existing workflows.

**Exercise 1: Working with the Azure Artifacts**

**Azure Artifacts** is an extension to Azure DevOps Services and Azure DevOps Server. It comes pre-installed in Azure DevOps Services, Azure DevOps Server 2019 and Team Foundation Server (TFS) 2017 and 2018.

***Task 1: Creating and connecting to a feed***

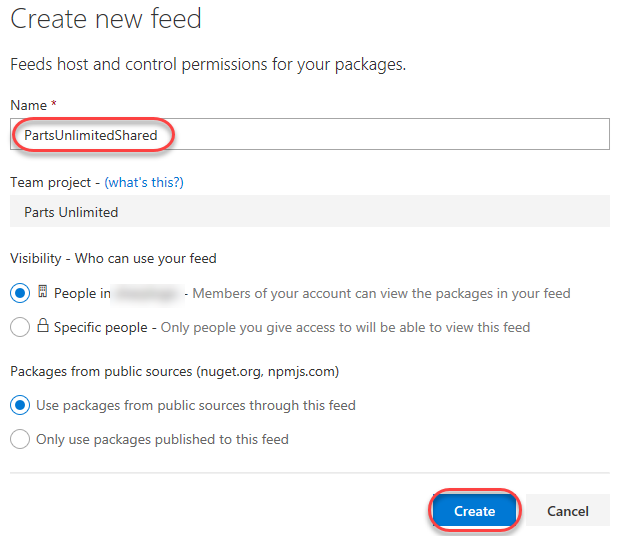
1. Navigate to the **Artifacts** hub.



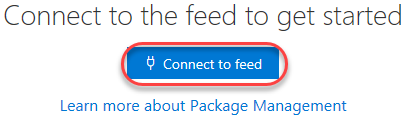
1. Click **New feed**. This feed will be a collection of NuGet packages available to users within the organization and will sit alongside the public NuGet feed as a peer. The scenario in this lab will focus on the workflow for using with the package management service, so the actual architectural and development decisions are purely illustrative.

https://azuredevopslabs.com/labs/azuredevops/packagemanagement/images/008.png

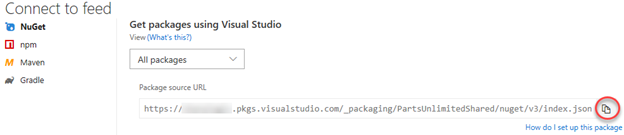
1. This feed will include common functionality that can be shared across projects in this organization. Set the name to **“PartsUnlimitedShared”**and click **Create**. Leave the default security options.



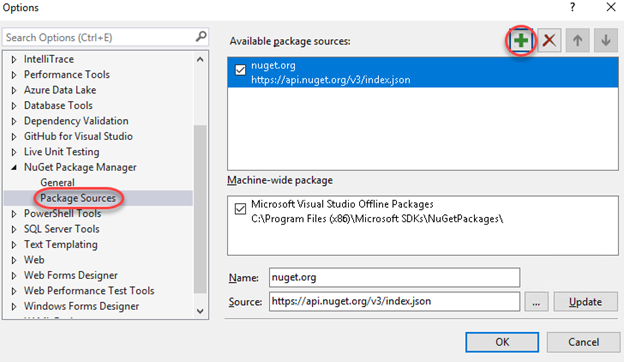
1. Any user who wants to connect to this NuGet feed must configure their environment. Click **Connect to feed**.



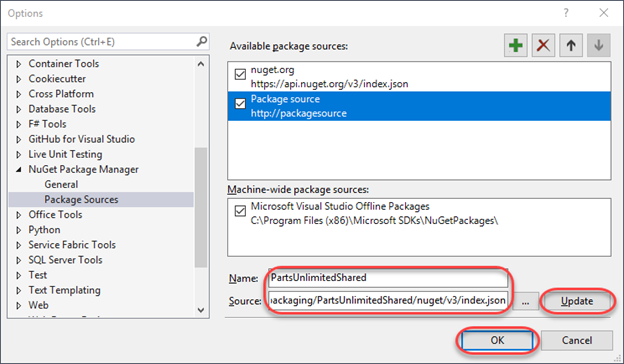
1. Copy the **Package source URL**. This is the only thing Visual Studio and NuGet need to start taking advantage of the new feed. Leave the dialog open in the browser.



1. Launch a new instance of **Visual Studio**. Do not use the instance open from cloning the primary **Parts Unlimited** solution.
2. Select **Tools | NuGet Package Manager | Package Manager Settings**.
3. Locate the **Package Sources** section and click the **Add** button to add a new package source.

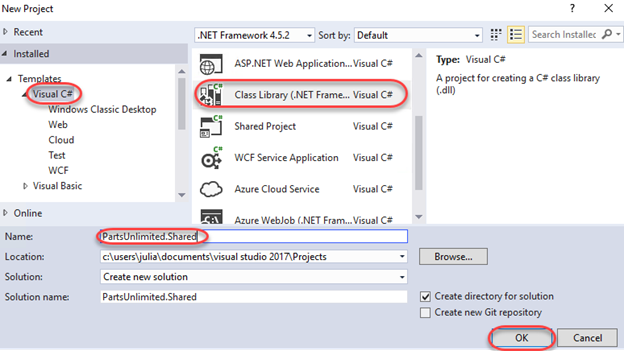


1. Set the **Name** to **“PartsUnlimitedShared”** and paste the **Source** URL copied earlier. Click **Update** followed by **OK**. Visual Studio is now connected to the new feed.

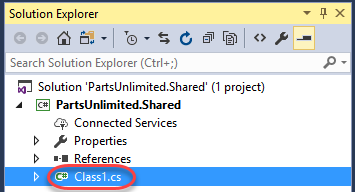


***Task 2: Creating and publishing a NuGet package***

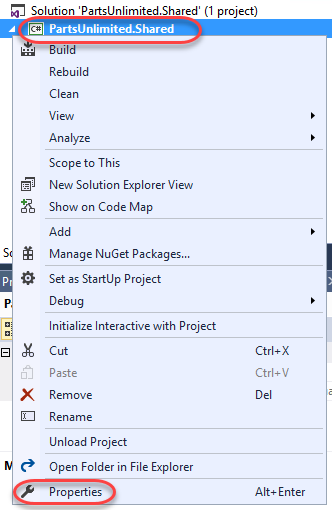
1. From the main menu of **Visual Studio**, select **File | New | Project**. We will now create a shared assembly that will be published as a NuGet package so that other teams can integrate it and stay up to date without having to work directly with the project source.
2. From the **Visual C#** section, select the **Class Library (.Net Framework)** template and set the **Name** to **“PartsUnlimited.Shared”**. Click **OK** to create the project.



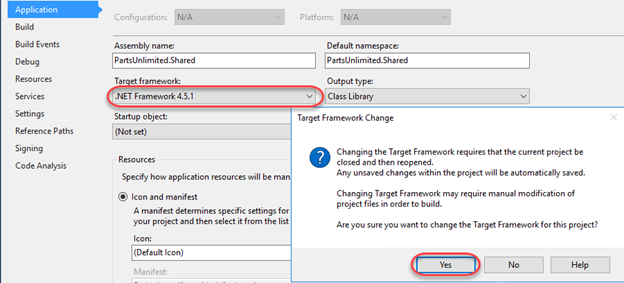
1. In **Solution Explorer**, delete **Class1.cs**.



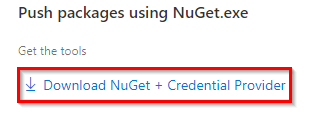
1. Right-click the project node and select **Properties**.



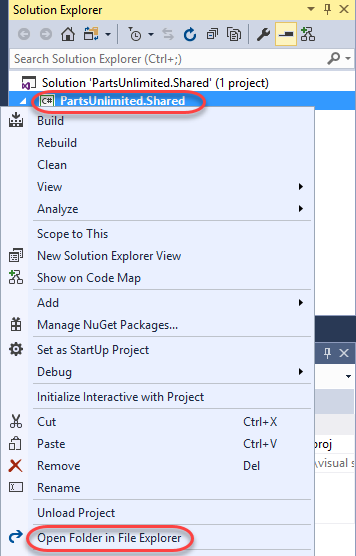
1. Set the **Target framework** to **.NET Framework 4.5.1** and click **Yes** to confirm the change.



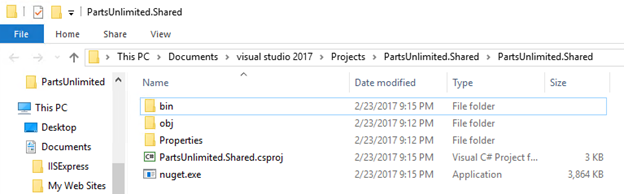
1. Press **Ctrl+Shift+B** to build the project. In the next task we’ll use **NuGet.exe** to generate a NuGet package directly from the built project, but it requires the project to be built first.
2. Return to the browser and click **Download NuGet + Credential Provider**. Extract the zip somewhere you can easily access.



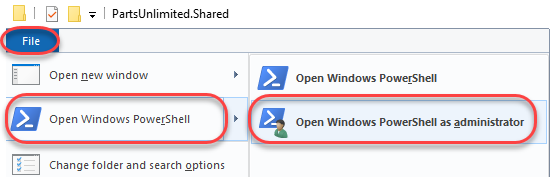
1. Return to **Visual Studio**. From **Solution Explorer**, right-click the **PartsUnlimited.Shared** project node and select **Open Folder in File Explorer**.



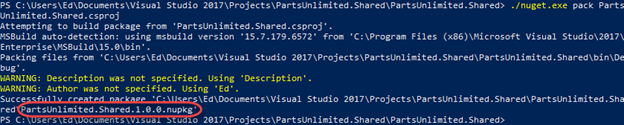
1. Move the downloaded **nuget.exe** and **CredentialProvider.VSS.exe** files into the folder containing the **.csproj** file.



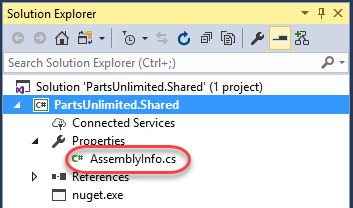
1. In the same **Windows Explorer** window, select **File | Open Windows PowerShell | Open Windows PowerShell as administrator**.



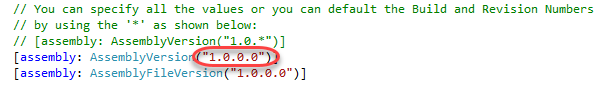
1. Execute the line below to create a **.nupkg** file from the project. Note that this is a quick shortcut to package the NuGet bits together for deployment. NuGet is very customizable and offers a lot of great flexibility for providing detailed information for consumers. You can learn more over on the [NuGet package creation page](https://docs.microsoft.com/en-us/nuget/create-packages/overview-and-workflowhttps:/docs.microsoft.com/en-us/nuget/create-packages/overview-and-workflow).
2. ./nuget.exe pack PartsUnlimited.Shared.csproj
3. NuGet builds a minimal package based on the information it is able to pull from the project. For example, note that the name is **PartsUnlimited.Shared.1.0.0.nupkg**. That version number was pulled from the assembly.



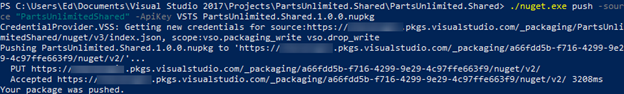
1. Return to **Visual Studio**. From **Solution Explorer**, open **Properties\AssemblyInfo.cs**.



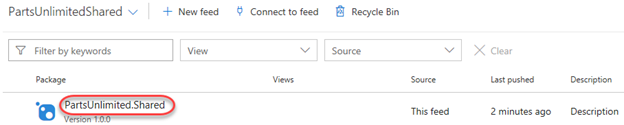
1. The **AssemblyVersion** attribute specifies the version number to build into the assembly. Each NuGet release requires a unique version number, so if we continue to use this method for creating packages, we will need to remember to increment this before building.



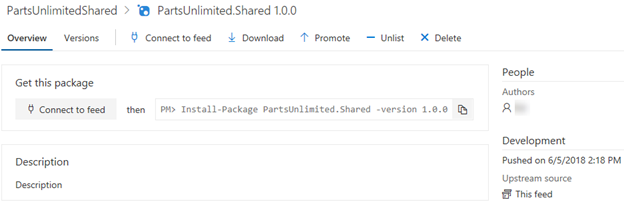
1. Return to the **PowerShell** window and execute the following command (it should be on one line). Note that you need to provide an “API Key”, which can be any non-empty string. We’re using **“VSTS”** here. Log in using your Azure DevOps credentials when asked.
2. ./nuget.exe push -source "PartsUnlimitedShared" -ApiKey VSTS PartsUnlimited.Shared.1.0.0.nupkg
3. The command should succeed after a few seconds.



1. Return to the browser window open to Azure DevOps and **refresh** the window. You should now see the organization’s NuGet package is published in the feed. Click it to view the details.

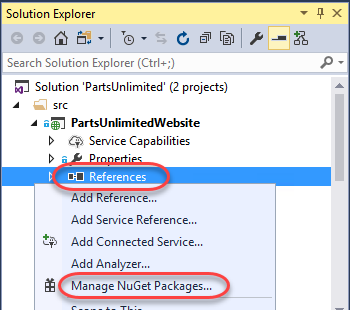


1. The details have been imported and are now ready for others to consume.

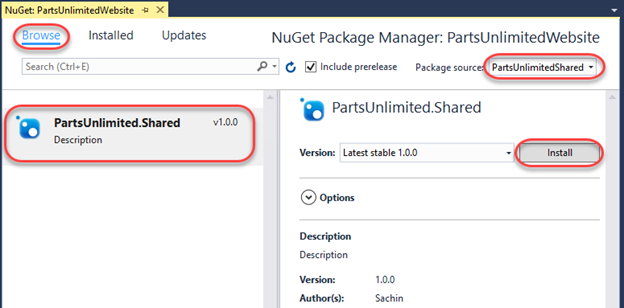


***Task 3: Importing a NuGet package***

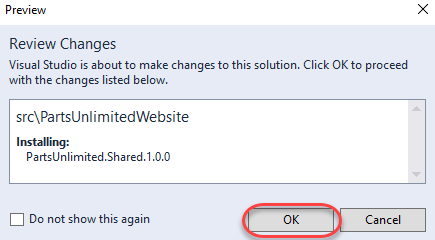
1. Switch to the instance of **Visual Studio** that has the full **Parts Unlimited** solution open.
2. In **Solution Explorer**, right-click the **References** node under the **PartsUnlimitedWebsite** project and select **Manage NuGet Packages**.



1. Click the **Browse** tab and change the **Package source** to **PartsUnlimitedShared**. The only package will be the one we just added, so click **Install** to add it to the project.



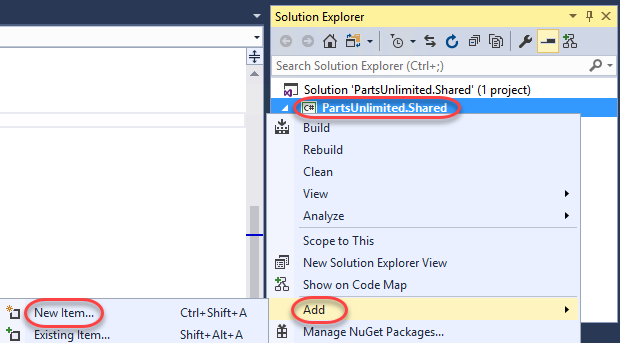
1. If asked, confirm the addition by clicking **OK**.



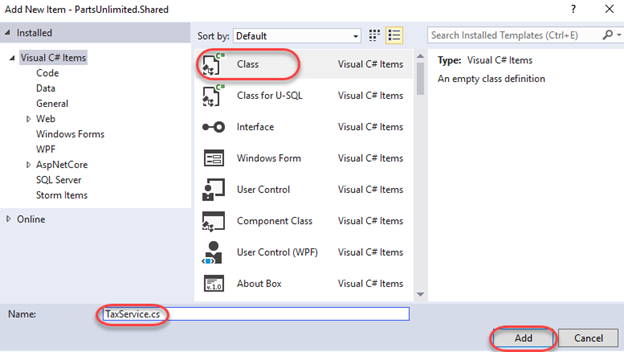
1. Press **Ctrl+Shift+B** to build the project. It should succeed. The NuGet package doesn’t add any value yet, but at least we know it’s there.

***Task 4: Updating a NuGet package***

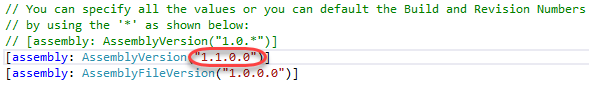
1. Switch to the instance of **Visual Studio** that has the **PartsUnlimited.Shared** project open (the NuGet source project).
2. In **Solution Explorer**, right-click the **PartsUnlimited.Shared** project node and select **Add | New Item**.



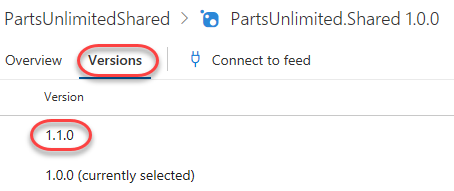
1. Select the **Class** template and enter the **Name “TaxService.cs”**. Click **Add** to add the class. We can pretend that tax calculation will be consolidated into this shared class and managed centrally so that other teams can simply work with the NuGet package.



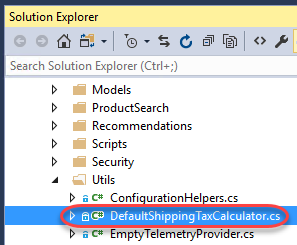
1. Replace the code in the new file with the code below. For now it will just hardcode a 10% rate.
2. namespace PartsUnlimited.Shared
3. {
4. public class TaxService
5. {
6. static public decimal CalculateTax(decimal taxable, string postalCode)
7. {
8. return taxable \* (decimal).1;
9. }
10. }
11. }
12. Since we’re updating the assembly (and package), return to **AssemblyInfo.cs** and update the **AssemblyVersion** to **1.1.0.0**.



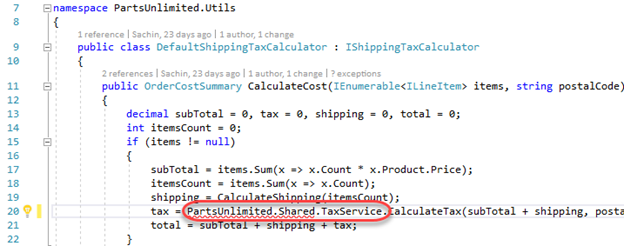
1. Press **Ctrl+Shift+B** to build the project.
2. Return to the **PowerShell** window and execute the following line to repackage the NuGet package. Note that the new package will have the updated version number.
3. ./nuget.exe pack PartsUnlimited.Shared.csproj
4. Execute the following line to publish the updated package. Note that the version number has changed to reflect the new package.
5. ./nuget.exe push -source "PartsUnlimitedShared" -ApiKey VSTS PartsUnlimited.Shared.1.1.0.nupkg
6. Return to the browser window open to Azure DevOps and refresh the page. You will still be looking at the 1.0.0 version of the package, but you can change that by selecting the **Versions** tab and selecting **1.1.0**.



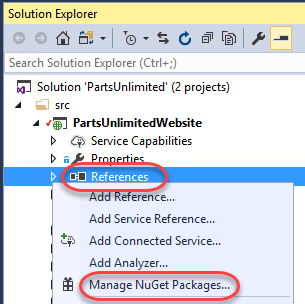
1. Switch back to the instance of **Visual Studio** open to the main **PartsUnlimited** project (with **PartsUnlimitedWebsite**).
2. From **Solution Explorer**, open **PartsUnlimitedWebsite\Utils\DefaultShippingTaxCalculator.cs**.



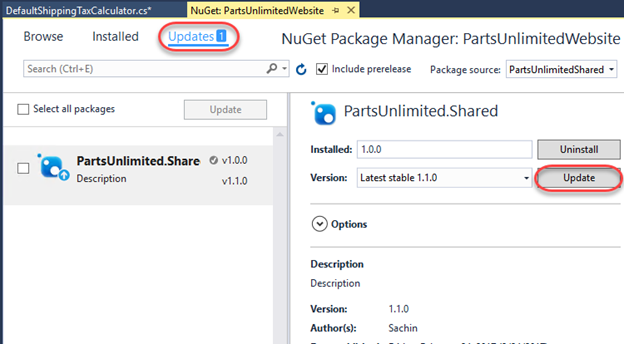
1. Locate the call to **CalculateTax** around line **20** and add the qualifier **“PartsUnlimited.Shared.TaxService.”** to the beginning. The original code called a method internal to this class, so the code we’re adding to the beginning of the line is redirecting it to code from our NuGet assembly. However, since this project hasn’t updated the NuGet package yet, it’s still referencing 1.0.0.0 and doesn’t have these new changes available, so the code will not build.



1. In **Solution Explorer**, right-click the **References** node and select **Manage NuGet Packages**.



1. NuGet is aware of our update, so click the **Updates** tab to view the details. Click **Update** to bring down the new version. If the **Updates** tab hasn’t yet updated, you can still update the package from the **Browse** tab. Note that there may be many NuGet updates available, but you should only need to update **PartsUnlimited.Shared**. Note that it may take a little while for the package to become completely available for updating. If you get an error, wait a moment and try again.



1. If asked, click **OK** to approve the update.
2. Press **F5** to build and run the site. It should work as expected.